PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: The ACM Digital Library The Guide

+provide +section +before +order +auxiliary overwrite overwi

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used
**provide section before order auxiliary overwrite overwritting**

Found **4,103** of **153,034**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Recipes for adjoint code construction

Ralf Giering, Thomas Kaminski

December 1998 **ACM Transactions on Mathematical Software (TOMS)**, Volume 24 Issue 4

Full text available: pdf(301.79 KB) Additional Information: full citation, abstract, references, citings, index terms

Adjoint models are increasingly being developed for use in meteorology and oceanography. Typical applications are data assimilation, model tuning, sensitivity analysis, and determination of singular vectors. The adjoint model computes the gradient of a cost function with respect to control variables. Generation of adjoint code may be seen as the special case of differentiation of algorithms in reverse mode, where the dependent function is a scalar. The described method for adjoint code gene ...

**Keywords**: adjoint model, adjoint operator, automatic differentiation, computational differentiation, data assimilation, differentiation of algorithms, implicit functions, inverse modeling, optimization, reverse mode

2 A software engineering perspective on algorithmics

Karsten Weihe

March 2001 **ACM Computing Surveys (CSUR)**, Volume 33 Issue 1

Full text available: pdf(1.62 MB) Additional Information: full citation, abstract, references, index terms, review

An algorithm component is an implementation of an algorithm which is not intended to be a stand-alone module, but to perform a specific task within a large software package or even within several distinct software packages. Therefore, the design of algorithm components must also incorporate software-engineering aspects. A key design goal is adaptability. This goal is important for maintenance throughout a project, prototypical development, and reuse in new, unforseen contex ...

**Keywords**: algorithm engineering

3 Formally based profiling for higher-order functional languages

Patrick M. Sansom, Simon L. Peyton Jones

March 1997 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 19 Issue 2

Full text available: pdf(551.43 KB) Additional Information: full citation, abstract, references, citings, index terms, review

We present the first source-level profiler for a compiled, nonstrict, higher-order, purely functional language capable of measuring time as well as space usage. Our profiler is

implemented in a production-quality optimizing compiler for Haskell and can successfully profile large applications. A unique feature of our approach is that we give a formal specification of the attribution of execution costs to cost centers. This specification enables us to discuss ...

**Keywords**: attribution of costs, cost centers, cost semantics, execution profiling, program transformation, source-level profiling, space profiling

4 Parallel execution of prolog programs: a survey

Gopal Gupta, Enrico Pontelli, Khayri A.M. Ali, Mats Carlsson, Manuel V. Hermenegildo

July 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 4

Full text available: pdf(1.95 MB) Additional Information: full citation, abstract, references, citings, index terms

Since the early days of logic programming, researchers in the field realized the potential for exploitation of parallelism present in the execution of logic programs. Their high-level nature, the presence of nondeterminism, and their referential transparency, among other characteristics, make logic programs interesting candidates for obtaining speedups through parallel execution. At the same time, the fact that the typical applications of logic programming frequently involve irregular computatio ...

**Keywords**: Automatic parallelization, constraint programming, logic programming, parallelism, prolog

5 Assembly instruction level reverse execution for debugging

Tankut Akgul, Vincent J. Mooney III

April 2004 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 13 Issue 2

Full text available: pdf(1.18 MB) Additional Information: full citation, abstract, references, index terms

Assembly instruction level reverse execution provides a programmer with the ability to return a program to a previous state in its execution history via execution of a "reverse program." The ability to execute a program in reverse is advantageous for shortening software development time. Conventional techniques for recovering a state rely on saving the state into a record before the state is destroyed. However, state-saving causes significant memory and time overheads during forward execution.Th ...

**Keywords**: Debugging, reverse code generation, reverse execution

6 Making parallel simulations go fast

Paul F. Reynolds, Carmen M. Pancerella, Sudhir Srinivasan

December 1992 **Proceedings of the 24th conference on Winter simulation**

Full text available: pdf(1.20 MB) Additional Information: full citation, references, citings, index terms

7 Live-structure dataflow analysis for Prolog

Anne Mulkers, William Winsborough, Maurice Bruynooghe

March 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 2

Full text available: pdf(3.59 MB) Additional Information: full citation, abstract, references, citings, index terms, review

For the class of applicative programming languages, efficient methods for reclaiming the memory occupied by released data structures constitute an important aspect of current implementations. The present article addresses the problem of memory reuse for logic programs through program analysis rather than by run-time garbage collection. The aim is to derive run-time properties that can be used at compile time to specialize the target code for a program according to a given set of queries and ...

**Keywords**: Prolog, abstract interpretation, compile-time garbage collection, liveness, program analysis

**8** The design of the E programming language

Joel E. Richardson, Michael J. Carey, Daniel T. Schuh

July 1993 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 15 Issue 3

Full text available: pdf(2.78 MB) Additional Information: full citation, references, citings, index terms, review

**Keywords**: extensible database systems, persistent object management

**9** Interprocedural slicing using dependence graphs

Susan Horwitz, Thomas Reps, David Binkley

January 1990 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 12 Issue 1

Full text available: pdf(2.69 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The notion of a program slice, originally introduced by Mark Weiser, is useful in program debugging, automatic parallelization, and program integration. A slice of a program is taken with respect to a program point p and a variable x; the slice consists of all statements of the program that might affect the value of x at point p. This paper concerns the problem of interprocedural slicing—generating a ...

**10** Prefetching in segmented disk cache for multi-disk systems CONSIDERED

Valery Soloviev

May 1996 **Proceedings of the fourth workshop on I/O in parallel and distributed systems: part of the federated computing research conference**

Full text available: pdf(1.57 MB) Additional Information: full citation, references, index terms

**11** Characterizations of Pushdown Machines in Terms of Time-Bounded Computers

Stephen A. Cook

January 1971 **Journal of the ACM (JACM)**, Volume 18 Issue 1

Full text available: pdf(1.20 MB) Additional Information: full citation, references, citings, index terms

**12** Techniques for reducing consistency-related communication in distributed shared-memory systems

John B. Carter, John K. Bennett, Willy Zwaenepoel

August 1995 **ACM Transactions on Computer Systems (TOCS)**, Volume 13 Issue 3

Full text available: pdf(2.86 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Distributed shared memory (DSM) is an abstraction of shared memory on a distributed-memory machine. Hardware DSM systems support this abstraction at the architecture level; software DSM systems support the abstraction within the runtime system. One of the key problems in building an efficient software DSM system is to reduce the amount of communication needed to keep the distributed memories consistent. In this article we present four techniques for doing so: software release consistency; m ...

**Keywords**: cache consistency protocols, distributed shared memory, memory models, release consistency, virtual shared memory

13 TransformGen: automating the maintenance of structure-oriented environments
David Garlan, Charles W. Krueger, Barbara Staudt Lerner
May 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 3

Full text available: pdf(3.10 MB) Additional Information: full citation, abstract, references, citings, index terms, review

A serious problem for programs that use persistent data is that information created and maintained by the program becomes invalid if the persistent types used in the program are modified in a new release. Unfortunately, there has been little systematic treatment of the problem; current approaches are manual, ad hoc, and time consuming both for programmers and users. In this article we present a new approach. Focusing on the special case of managing abstract syntax trees in structure-oriente ...

**Keywords**: schema evolution, structure-oriented environments, type evolution

14 Quorum consensus in nested-transaction systems
Kenneth J. Goldman, Nancy Lynch
December 1994 **ACM Transactions on Database Systems (TODS)**, Volume 19 Issue 4

Full text available: pdf(3.45 MB) Additional Information: full citation, abstract, references, index terms

Gifford's Quorum Consensus algorithm for data replication is studied in the context of nested transactions and transaction failures (aborts), and a fully developed reconfiguration strategy is presented. A formal description of the algorithm is presented using the Input/Output automaton model for nested-transaction systems due to Lynch and Merritt. In this description, the algorithm itself is described in terms of nested transactions. The formal description is used to construct a complete pr ...

**Keywords**: I/O automata, concurrency control, data replication, hierarchical proofs, nested transactions, quorum consensus

15 A practical approach to multiple default inheritance for unification-based lexicons
Graham Russell, Afzal Ballim, John Carroll, Susan Warwick-Armstrong
September 1992 **Computational Linguistics**, Volume 18 Issue 3

Full text available: pdf(1.86 MB) Additional Information: full citation, abstract, references, citings
Publisher Site

This paper describes a unification-based lexicon system for NLP applications that incorporates mechanisms for multiple default inheritance. Such systems are intractable in the general case---the approach adopted here places a number of restrictions on the inheritance hierarchy in order to remove some of the sources of complexity while retaining more desirable properties. Implications of the design choices are discussed, comparisons are drawn with related work in computational linguistics and AI, ...

16 Wait-free synchronization
Maurice Herlihy
January 1991 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 13 Issue 1

Full text available: pdf(1.75 MB) Additional Information: full citation, abstract, references, citings, index terms, review

A wait-free implementation of a concurrent data object is one that guarantees that any process can complete any operation in a finite number of steps, regardless of the execution speeds of the other processes. The problem of constructing a wait-free implementation of one data object from another lies at the heart of much recent work in concurrent algorithms, concurrent data structures, and multiprocessor architectures. First, we introduce a simple and general technique, bas ...

**Keywords**: linearization, wait-free synchronization

**17** Making a fast curry: push/enter vs. eval/apply for higher-order languages
Simon Marlow, Simon Peyton Jones
September 2004 **ACM SIGPLAN Notices , Proceedings of the ninth ACM SIGPLAN international conference on Functional programming**, Volume 39 Issue 9

Full text available: pdf(186.19 KB) Additional Information: full citation, abstract, references, citings, index terms

Higher-order languages that encourage currying are implemented using one of two basic evaluation models: push/enter or eval/apply. Implementors use their intuition and qualitative judgements to choose one model or the other.Our goal in this paper is to provide, for the first time, a more substantial basis for this choice, based on our qualitative and quantitative experience of implementing both models in a state-of-the-art compiler for Haskell.Our conclusion is simple, and contradicts our initia ...

**18** Continuous program optimization: A case study
Thomas Kistler, Michael Franz
July 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 4

Full text available: pdf(877.67 KB) Additional Information: full citation, abstract, references, index terms, review

Much of the software in everyday operation is not making optimal use of the hardware on which it actually runs. Among the reasons for this discrepancy are hardware/software mismatches, modularization overheads introduced by software engineering considerations, and the inability of systems to adapt to users' behaviors.A solution to these problems is to delay code generation until load time. This is the earliest point at which a piece of software can be fine-tuned to the actual capabilities of the ...

**Keywords**: Dynamic code generation, continuous program optimization, dynamic reoptimization

**19** The design and implementation of a log-structured file system
Mendel Rosenblum, John K. Ousterhout
February 1992 **ACM Transactions on Computer Systems (TOCS)**, Volume 10 Issue 1

Full text available: pdf(1.97 MB) Additional Information: full citation, abstract, references, citings, index terms, review

This paper presents a new technique for disk storage management called a log-structured file system. A log-structured file system writes all modifications to disk sequentially in a log-like structure, thereby speeding up both file writing and crash recovery. The log is the only structure on disk; it contains indexing information so that files can be read back from the log efficiently. In order to maintain large free areas on disk for fast writing, we divide the log into
**Keywords**: Unix, disk storage management, fast crash recovery, file system organization, file system performance, high write performance, log-structured, logging

**20** Simple and efficient bounded concurrent timestamping or bounded concurrent timestamp systems are comprehensible!
Cynthia Dwork, Orli Waarts
July 1992 **Proceedings of the twenty-fourth annual ACM symposium on Theory of computing**

Full text available: pdf(1.19 MB) Additional Information: full citation, references, citings, index terms, review



The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: The ACM Digital Library The Guide

+provide +main +auxiliary +book +content

SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **provide main auxiliary book content** Found **481** of **153,034**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

**1** Third Generation Computer Systems

Peter J. Denning

December 1971 **ACM Computing Surveys (CSUR)**, Volume 3 Issue 4

Full text available: pdf(3.52 MB) Additional Information: full citation, abstract, references, citings, index terms

The common features of third generation operating systems are surveyed from a general view, with emphasis on the common abstractions that constitute at least the basis for a "theory" of operating systems. Properties of specific systems are not discussed except where examples are useful. The technical aspects of issues and concepts are stressed, the nontechnical aspects mentioned only briefly. A perfunctory knowledge of third generation systems is presumed.

**2** Shared books: collaborative publication management for an office information system

Brian T. Lewis, Jeffrey D. Hodges

April 1988 **ACM SIGOIS Bulletin , Conference Sponsored by ACM SIGOIS and IEEECS TC-OA on Office information systems**, Volume 9 Issue 2-3

Full text available: pdf(1.10 MB) Additional Information: full citation, abstract, references, citings, index terms

A Shared Book helps the users of an office information system create a multiple-part publication and manage it throughout its life cycle. The Shared Book supports simultaneous collaboration both by allowing different workers to work on different parts at the same time and by ensuring that workers use the current revision of each part. It protects publication information by providing locking and access control. The Shared Book communicates the publication's current state to ...

CONSIDERED

**3** DIAGRAM: a grammar for dialogues

Jane J. Robinson

January 1982 **Communications of the ACM**, Volume 25 Issue 1

Full text available: pdf(2.11 MB) Additional Information: full citation, abstract, references, citings, index terms

An explanatory overview is given of DIAGRAM, a large and complex grammar used in an artificial intelligence system for interpreting English dialogue. DIAGRAM is an augmented phrase-structure grammar with rule procedures that allow phrases to inherit attributes from their constituents and to acquire attributes from the larger phrases in which they themselves are constituents. These attributes are used to set context-sensitive constraints on the acceptance of an analysis. Constraints can be i ...

**Keywords**: annotations, attribute inheritance, augmented rules, contextual constraints, dialogue, likelihoods, metarules, phrase-structure grammar, transformations

4 Books and reading: Realistic books: a bizarre homage to an obsolete medium?
Yi-Chun Chu, David Bainbridge, Matt Jones, Ian H. Witten
June 2004 **Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries**

Full text available: pdf(2.24 MB) Additional Information: full citation, abstract, references, index terms

For many readers, handling a physical book is an enjoyably exquisite part of the information seeking process. Many physical characteristics of a book-its size, heft, the patina of use on its pages and so on-communicate ambient qualities of the document it represents. In contrast, the experience of accessing and exploring digital library documents is often dull. The emphasis is utilitarian; technophile rather than bibliophile. We have extended the page-turning algorithm we reported at last year's ...

**Keywords**: 3D book visualisation, Java and OpenGL, visual metadata

5 Curriculum 68: Recommendations for academic programs in computer science: a report of the ACM curriculum committee on computer science
William F. Atchison, Samuel D. Conte, John W. Hamblen, Thomas E. Hull, Thomas A. Keenan, William B. Kehl, Edward J. McCluskey, Silvio O. Navarro, Werner C. Rheinboldt, Earl J. Schweppe, William Viavant, David M. Young
March 1968 **Communications of the ACM**, Volume 11 Issue 3

Full text available: pdf(6.63 MB) Additional Information: full citation, references, citings

**Keywords**: computer science academic programs, computer science bibliographies, computer science courses, computer science curriculum, computer science education, computer science graduate programs, computer science undergraduate programs

6 Fast detection of communication patterns in distributed executions
Thomas Kunz, Michiel F. H. Seuren
November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**

Full text available: pdf(4.21 MB) Additional Information: full citation, abstract, references, index terms

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagrams are often used to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an event tracer developed at the University of Waterloo. However, these diagrams are often very complex and do not provide the user with the desired overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun ...

7 Automatic parsing for content analysis
Frederick J. Damerau
June 1970 **Communications of the ACM**, Volume 13 Issue 6

Full text available: pdf(4.07 MB) Additional Information: full citation, abstract, references, citings

Although automatic syntactic and semantic analysis is not yet possible for all of an unrestricted natural language text, some applications, of which content analysis is one, do not have such a stringent coverage requirement. Preliminary studies show that the Harvard Syntactic Analyzer can produce correct and unambiguous identification of the subject and object of certain verbs for approximately half of the relevant occurences. This provides a degree of coverage for content analysis variable ...

**Keywords**: content analysis, information retrieval, language analysis, natural language processing, parsing, syntactic analysis, text processing

8 Curriculum recommendations for graduate professional programs in information systems

May 1972 **Communications of the ACM**, Volume 15 Issue 5

Full text available: pdf(4.00 MB) Additional Information: full citation, references, citings

**Keywords**: education, information analysis, information systems development, management information systems, management systems, system design, systems analysis

9 Virtual Memory

Peter J. Denning

September 1970 **ACM Computing Surveys (CSUR)**, Volume 2 Issue 3

Full text available: pdf(2.63 MB) Additional Information: full citation, references, citings, index terms

10 Curriculum recommendations for undergraduate programs in information systems

J. Daniel Couger

December 1973 **Communications of the ACM**, Volume 16 Issue 12

Full text available: pdf(3.23 MB) Additional Information: full citation, references, citings

**Keywords**: education, information analysis, information systems, management systems, systems analysis, systems design, undergraduate curricula

11 A survey of approaches to automatic schema matching

Erhard Rahm, Philip A. Bernstein

December 2001 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 10 Issue 4

Full text available: pdf(196.22 KB) Additional Information: full citation, abstract, citings, index terms

Schema matching is a basic problem in many database application domains, such as data integration, E-business, data warehousing, and semantic query processing. In current implementations, schema matching is typically performed manually, which has significant limitations. On the other hand, previous research papers have proposed many techniques to achieve a partial automation of the match operation for specific application domains. We present a taxonomy that covers many of these existing approach ...

**Keywords**: Graph matching, Machine learning, Model management, Schema integration, Schema matching

12 Associative Processor Architecture—a Survey

S. S. Yau, H. S. Fung

January 1977 **ACM Computing Surveys (CSUR)**, Volume 9 Issue 1

Full text available: pdf(1.87 MB) Additional Information: full citation, references, citings, index terms

13 Book Reviews: The book review column

William Gasarch

March 2002 **ACM SIGACT News**, Volume 33 Issue 1

Full text available: pdf(1.34 MB) Additional Information: full citation

14 Human-computer interface development: concepts and systems for its management

H. Rex Hartson, Deborah Hix

March 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 1

Full text available: pdf(7.97 MB) Additional Information: full citation, abstract, references, citings, index terms, review

*Human-computer interface management*, from a computer science viewpoint, focuses on the process of developing quality human-computer interfaces, including their representation, design, implementation, execution, evaluation, and maintenance. This survey presents important concepts of interface management: dialogue independence, structural modeling, representation, interactive tools, rapid prototyping, development methodologies, and control structures. *Dialogue independence* is th ...

**15** Garbage Collection of Linked Data Structures
Jacques Cohen
September 1981 **ACM Computing Surveys (CSUR)**, Volume 13 Issue 3

Full text available: pdf(2.32 MB) Additional Information: full citation, references, citings, index terms

**16** Associative and Parallel Processors
Kenneth J. Thurber, Leon D. Wald
December 1975 **ACM Computing Surveys (CSUR)**, Volume 7 Issue 4

Full text available: pdf(2.62 MB) Additional Information: full citation, references, citings, index terms

**17** Spoken dialogue technology: enabling the conversational user interface
Michael F. McTear
March 2002 **ACM Computing Surveys (CSUR)**, Volume 34 Issue 1

Full text available: pdf(987.69 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Spoken dialogue systems allow users to interact with computer-based applications such as databases and expert systems by using natural spoken language. The origins of spoken dialogue systems can be traced back to Artificial Intelligence research in the 1950s concerned with developing conversational interfaces. However, it is only within the last decade or so, with major advances in speech technology, that large-scale working systems have been developed and, in some cases, introduced into commerc ...

**Keywords**: Dialogue management, human computer interaction, language generation, language understanding, speech recognition, speech synthesis

**18** Fortran 8X draft
Loren P. Meissner
December 1989 **ACM SIGPLAN Fortran Forum**, Volume 8 Issue 4

Full text available: pdf(21.36 MB) Additional Information: full citation, abstract, index terms

**Standard Programming Language Fortran.** This standard specifies the form and establishes the interpretation of programs expressed in the Fortran language. It consists of the specification of the language Fortran. No subsets are specified in this standard. The previous standard, commonly known as "FORTRAN 77", is entirely contained within this standard, known as "Fortran 8x". Therefore, any standard-conforming FORTRAN 77 program is standard conforming under this standard. New features can b ...

**19** Computer Software and Copyright
Calvin N. Mooers
January 1975 **ACM Computing Surveys (CSUR)**, Volume 7 Issue 1

Full text available: pdf(2.63 MB) Additional Information: full citation, references, citings, index terms

**20** Incremental validation of XML documents

Andrey Balmin, Yannis Papakonstantinou, Victor Vianu
December 2004 **ACM Transactions on Database Systems (TODS)**, Volume 29 Issue 4

Full text available: pdf(676.95 KB) Additional Information: full citation, abstract, references, index terms

We investigate the incremental validation of XML documents with respect to DTDs, specialized DTDs, and XML Schemas, under updates consisting of element tag renamings, insertions, and deletions. DTDs are modeled as extended context-free grammars. "Specialized DTDs" allow the decoupling of element types from element tags. XML Schemas are abstracted as specialized DTDs with limitations on the type assignment. For DTDs and XML Schemas, we exhibit an $O(m \log n)$ incremental valida ...

**Keywords**: Update, XML, validation

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: The ACM Digital Library The Guide

+provide +section +before overwrite overwritting

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **provide section before overwrite overwritting** Found **53,386** of **153,034**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 A coherent distributed file cache with directory write-behind

Timothy Mann, Andrew Birrell, Andy Hisgen, Charles Jerian, Garret Swart

May 1994 **ACM Transactions on Computer Systems (TOCS)**, Volume 12 Issue 2

Full text available: pdf(3.21 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Extensive caching is a key feature of the Echo distributed file system. Echo client machines maintain coherent caches of file and directory data and properties, with write-behind (delayed write-back) of all cached information. Echo specifies ordering constraints on this write-behind, enabling applications to store and maintain consistent data structures in the file system even when crashes or network faults prevent some writes from being completed. In this paper we describe ...

**Keywords**: coherence, file caching, write-behind

CONSIDERED

2 Garbage collection for a client-server persistent object store

Laurent Amsaleg, Michael J. Franklin, Olivier Gruber

August 1999 **ACM Transactions on Computer Systems (TOCS)**, Volume 17 Issue 3

Full text available: pdf(267.18 KB) Additional Information: full citation, abstract, references, citings, index terms, review

We describe an efficient server-based algorithm for garbage collecting persistent object stores in a client-server environmnet. The algorithm is incremental and runs concurrently with client transactions. Unlike previous algorithms, it does not hold any transactional locks on data and does non require callbacks to clients. It is fault-tolerant, but performs very little logging. The algorithm has been designed to be integrated into existing systems, and therefore it works with standard i ...

**Keywords**: client-server system, logging, persistent object-store, recovery

3 FLAME: Formal Linear Algebra Methods Environment

John A. Gunnels, Fred G. Gustavson, Greg M. Henry, Robert A. van de Geijn

December 2001 **ACM Transactions on Mathematical Software (TOMS)**, Volume 27 Issue 4

Full text available: pdf(508.85 KB) Additional Information: full citation, abstract, references, citings, index terms

Since the advent of high-performance distributed-memory parallel computing, the need for intelligible code has become ever greater. The development and maintenance of libraries for these architectures is simply too complex to be amenable to conventional approaches to implementation. Attempts to employ traditional methodology have led, in our opinion, to the production of an abundance of anfractuous code that is difficult to maintain and almost impossible to upgrade.Having struggled with these is ...

**Keywords**: Formal derivation, libraries, linear algebra, performance

4 File and storage systems: The Google file system
Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung
October 2003 **Proceedings of the nineteenth ACM symposium on Operating systems principles**
Full text available: pdf(275.54 KB) Additional Information: full citation, references, index terms

**Keywords**: clustered storage, data storage, fault tolerance, scalability

5 File servers for network-based distributed systems
Liba Svobodova
December 1984 **ACM Computing Surveys (CSUR)**, Volume 16 Issue 4
Full text available: pdf(4.23 MB) Additional Information: full citation, references, citings, index terms, review

6 TransformGen: automating the maintenance of structure-oriented environments
David Garlan, Charles W. Krueger, Barbara Staudt Lerner
May 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 3
Full text available: pdf(3.10 MB) Additional Information: full citation, abstract, references, citings, index terms, review

A serious problem for programs that use persistent data is that information created and maintained by the program becomes invalid if the persistent types used in the program are modified in a new release. Unfortunately, there has been little systematic treatment of the problem; current approaches are manual, ad hoc, and time consuming both for programmers and users. In this article we present a new approach. Focusing on the special case of managing abstract syntax trees in structure-oriente ...

**Keywords**: schema evolution, structure-oriented environments, type evolution

7 Hardware and Binary Modification Support for Code Pointer Protection From Buffer Overflow
Nathan Tuck, Brad Calder, George Varghese
December 2004 **Proceedings of the 37th International Symposium on Microarchitecture**
Full text available: pdf(294.15 KB) Additional Information: full citation, abstract

Buffer overflow vulnerabilities are currently the most prevalent security vulnerability; they are responsible for over half of the CERT advisories issued in the last three years.Since many attacks exploit buffer overflow vulnerabilities, techniques that prevent buffer overflow attacks would greatly increase the difficulty of writing a new worm. This paper examines both software and hardware solutions for protecting code pointers from buffer overflow attacks.We first evaluate the performance over ...

8 The design, implementation, and evaluation of Jade
Martin C. Rinard, Monica S. Lam
May 1998 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 20 Issue 3
Full text available: pdf(576.88 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Jade is a portable, implicitly parallel language designed for exploiting task-level concurrency.Jade programmers start with a program written in a standard serial, imperative language, then use Jade constructs to declare how parts of the program access data. The

Jade implementation uses this data access information to automatically extract the concurrency and map the application onto the machine at hand. The resulting parallel execution preserves the semantics of the original serial program ...

**Keywords**: parallel computing, parallel programming languages

**9** Parallel execution of prolog programs: a survey

Gopal Gupta, Enrico Pontelli, Khayri A.M. Ali, Mats Carlsson, Manuel V. Hermenegildo

July 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 4

Full text available: pdf(1.95 MB) Additional Information: full citation, abstract, references, citings, index terms

Since the early days of logic programming, researchers in the field realized the potential for exploitation of parallelism present in the execution of logic programs. Their high-level nature, the presence of nondeterminism, and their referential transparency, among other characteristics, make logic programs interesting candidates for obtaining speedups through parallel execution. At the same time, the fact that the typical applications of logic programming frequently involve irregular computatio ...

**Keywords**: Automatic parallelization, constraint programming, logic programming, parallelism, prolog

**10** Prefetching in segmented disk cache for multi-disk systems

Valery Soloviev

May 1996 **Proceedings of the fourth workshop on I/O in parallel and distributed systems: part of the federated computing research conference**

Full text available: pdf(1.57 MB) Additional Information: full citation, references, index terms

**11** Consistency management for virtually indexed caches

Bob Wheeler, Brian N. Bershad

September 1992 **ACM SIGPLAN Notices , Proceedings of the fifth international conference on Architectural support for programming languages and operating systems**, Volume 27 Issue 9

Full text available: pdf(1.62 MB) Additional Information: full citation, references, citings, index terms

**12** Assembly instruction level reverse execution for debugging

Tankut Akgul, Vincent J. Mooney III

April 2004 **ACM Transactions on Software Engineering and Methodology (TOSEM)**, Volume 13 Issue 2

Full text available: pdf(1.18 MB) Additional Information: full citation, abstract, references, index terms

Assembly instruction level reverse execution provides a programmer with the ability to return a program to a previous state in its execution history via execution of a "reverse program." The ability to execute a program in reverse is advantageous for shortening software development time. Conventional techniques for recovering a state rely on saving the state into a record before the state is destroyed. However, state-saving causes significant memory and time overheads during forward execution.Th ...

**Keywords**: Debugging, reverse code generation, reverse execution

**13** Randomized instruction set emulation

Elena Gabriela Barrantes, David H. Ackley, Stephanie Forrest, Darko Stefanović

February 2005 **ACM Transactions on Information and System Security (TISSEC)**, Volume 8 Issue 1

Full text available: Additional Information:

pdf(374.44 KB) full citation, abstract, references, index terms

Injecting binary code into a running program is a common form of attack. Most defenses employ a "guard the doors" approach, blocking known mechanisms of code injection. *Randomized instruction set emulation* (RISE) is a complementary method of defense, one that performs a hidden randomization of an application's machine code. If foreign binary code is injected into a program running under RISE, it will not be executable because it will not know the proper randomization. The pape ...

**Keywords**: Automated diversity, randomized instruction sets, software diversity

**14** Extended ephemeral logging: log storage management for applications with long lived transactions

John S. Keen, William J. Dally

March 1997 **ACM Transactions on Database Systems (TODS)**, Volume 22 Issue 1

Full text available: pdf(566.34 KB) Additional Information: full citation, references, index terms, review

**Keywords**: OLTP, disk management, logging, long transactions

**15** RAID: high-performance, reliable secondary storage

Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, David A. Patterson

June 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 2

Full text available: pdf(3.60 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Disk arrays were proposed in the 1980s as a way to use parallelism between multiple disks to improve aggregate I/O performance. Today they appear in the product lines of most major computer manufacturers. This article gives a comprehensive overview of disk arrays and provides a framework in which to organize current and future work. First, the article introduces disk technology and reviews the driving forces that have popularized disk arrays: performance and reliability. It discusses the tw ...

**Keywords**: RAID, disk array, parallel I/O, redundancy, storage, striping

**16** Special issue on persistent object systems: Orthogonally persistent object systems

Malcolm Atkinson, Ronald Morrison

July 1995 **The VLDB Journal — The International Journal on Very Large Data Bases**, Volume 4 Issue 3

Full text available: pdf(5.02 MB) Additional Information: full citation, abstract, references, citings

Persistent Application Systems (PASs) are of increasing social and economic importance. They have the potential to be long-lived, concurrently accessed, and consist of large bodies of data and programs. Typical examples of PASs are CAD/CAM systems, office automation, CASE tools, software engineering environments, and patient-care support systems in hospitals. Orthogonally persistent object systems are intended to provide improved support for the design, construction, maintenance, and operation o ...

**Keywords**: database programming languages, orthogonal persistence, persistent application systems, persistent programming languages

**17** Data-Driven and Demand-Driven Computer Architecture

Philip C. Treleaven, David R. Brownbridge, Richard P. Hopkins

January 1982 **ACM Computing Surveys (CSUR)**, Volume 14 Issue 1

Full text available: pdf(4.14 MB) Additional Information: full citation, references, citings, index terms

18 Secure program execution via dynamic information flow tracking

G. Edward Suh, Jae W. Lee, David Zhang, Srinivas Devadas

October 2004 **Proceedings of the 11th international conference on Architectural support for programming languages and operating systems**, Volume 39 , 38 , 32 Issue 11 , 5 , 5

Full text available: pdf(263.33 KB) Additional Information: full citation, abstract, references, index terms

We present a simple architectural mechanism called dynamic information flow tracking that can significantly improve the security of computing systems with negligible performance overhead. Dynamic information flow tracking protects programs against malicious software attacks by identifying spurious information flows from untrusted I/O and restricting the usage of the spurious information.Every security attack to take control of a program needs to transfer the program's control to malevolent code. ...

**Keywords**: buffer overflow, format string, hardware tagging

19 Implementing crash recovery in QuickStore: a performance study

Seth J. White, David J. DeWitt

May 1995 **ACM SIGMOD Record , Proceedings of the 1995 ACM SIGMOD international conference on Management of data**, Volume 24 Issue 2

Full text available: pdf(1.67 MB) Additional Information: full citation, abstract, references, citings, index terms

Implementing crash recovery in an Object-Oriented Database System (OODBMS) raises several challenging issues for performance that are not present in traditional DBMSs. These performance concerns result both from significant architectural differences between OODBMSs and traditional database systems and differences in OODBMS's target applications. This paper compares the performance of several alternative approaches to implementing crash recovery in an OODBMS based on a client-server architecture. ...

20 Three-dimensional medical imaging: algorithms and computer systems

M. R. Stytz, G. Frieder, O. Frieder

December 1991 **ACM Computing Surveys (CSUR)**, Volume 23 Issue 4

Full text available: pdf(7.38 MB) Additional Information: full citation, references, citings, index terms, review

**Keywords**: Computer graphics, medical imaging, surface rendering, three-dimensional imaging, volume rendering



The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player